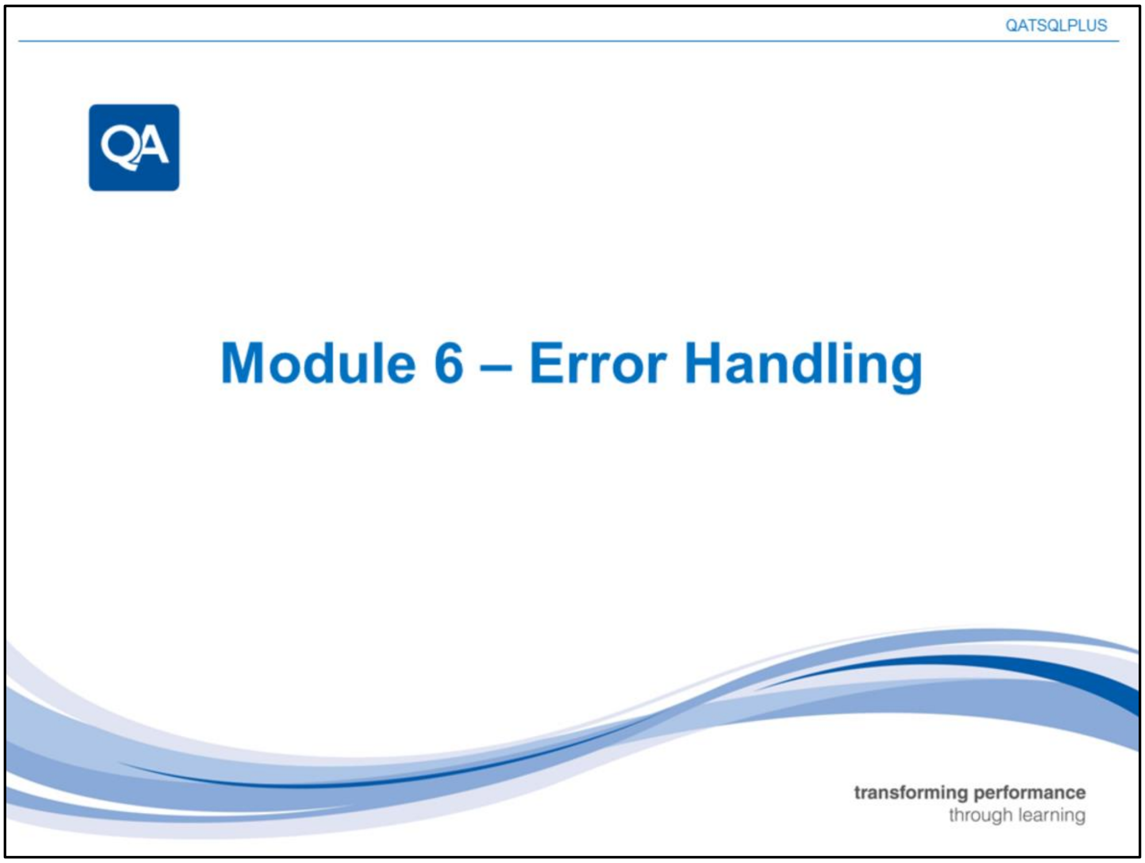
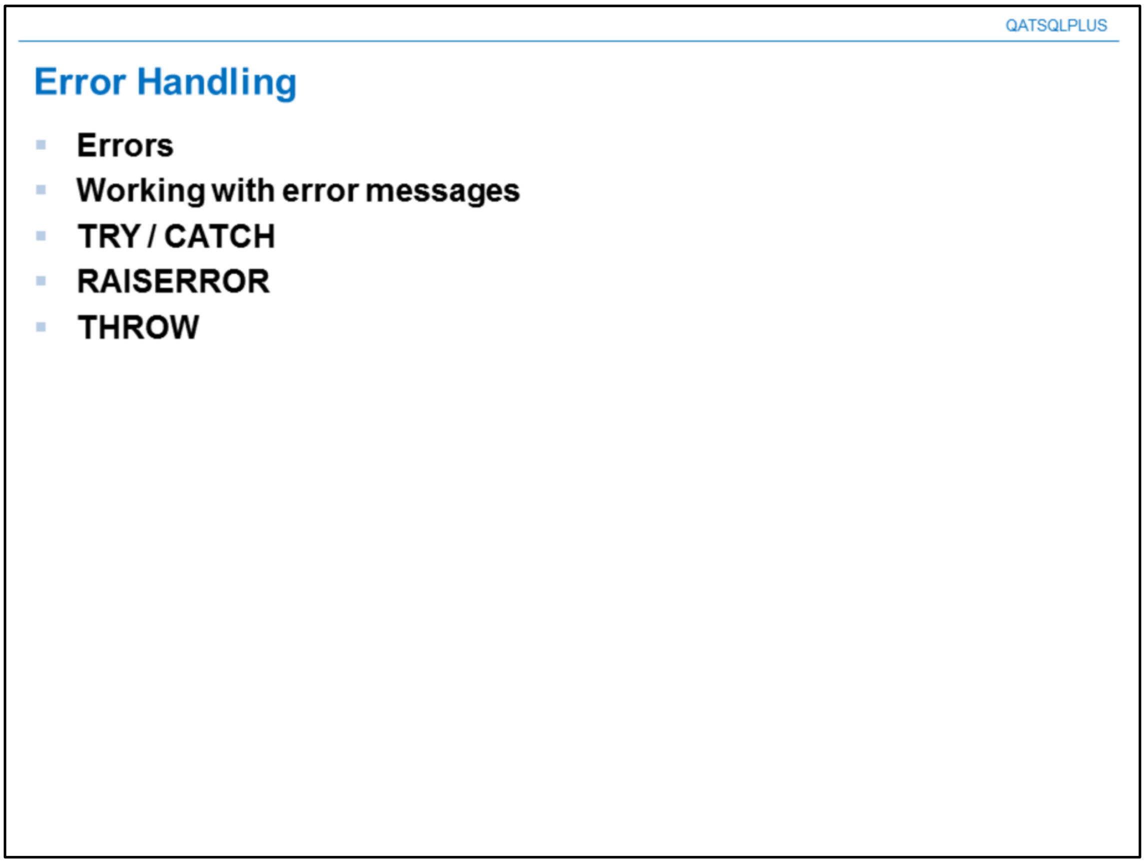


QATSQLPLUS

Module 6 – Error Handling

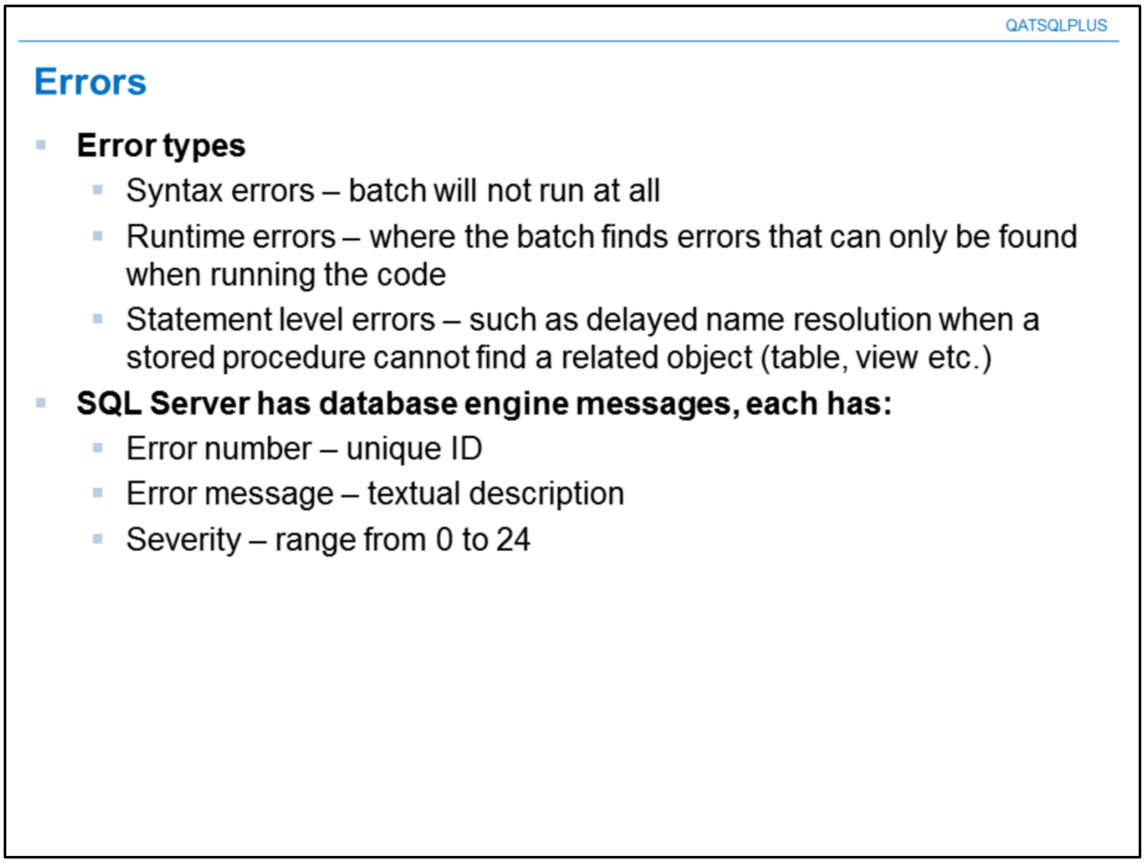
transforming performance
through learning



QATSQLPLUS

Error Handling

- Errors
- Working with error messages
- TRY / CATCH
- RAISERROR
- THROW

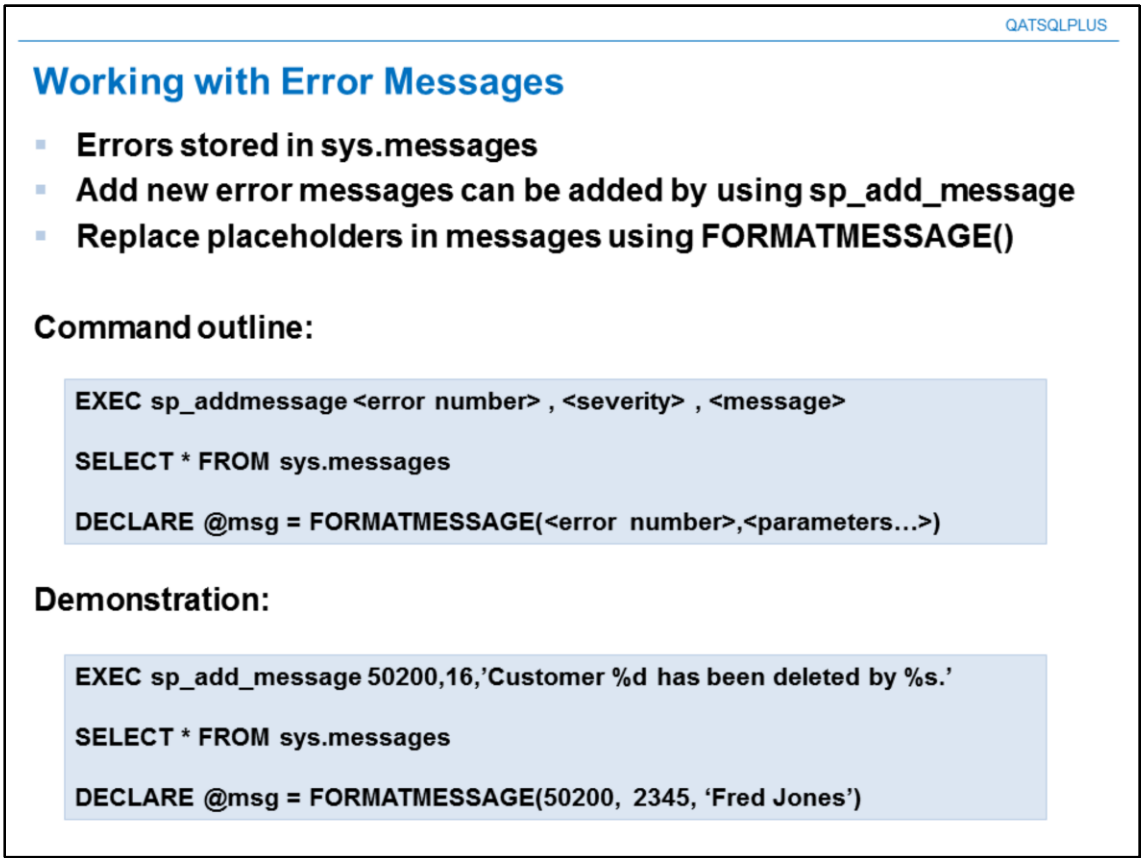

QATSQLPLUS

Errors

- **Error types**
  - Syntax errors – batch will not run at all
  - Runtime errors – where the batch finds errors that can only be found when running the code
  - Statement level errors – such as delayed name resolution when a stored procedure cannot find a related object (table, view etc.)
- **SQL Server has database engine messages, each has:**
  - Error number – unique ID
  - Error message – textual description
  - Severity – range from 0 to 24

Errors

- Error types
  - **Syntax errors** – batch will not run at all, but subsequent batches will execute
  - **Runtime errors** – where the code finds errors that can only be found when running the code
    - Explicit data type conversions where the value cannot be converted, such as a date held as a string to a date data type, where the date is in this wrong format
  - **Statement level errors**
    - Delayed name resolution when a stored procedure cannot find a related object (table, view, etc.)
- SQL Server has database engine messages, each has:
  - Error number – unique ID
  - Error message – textual description
  - Severity
    - 0-10: Informational messages and error that are not severe.
    - 11-16: Errors that can be corrected by the user (message shown in red)
    - 17-19: Software errors that cannot be corrected by the user, and they need to inform the administrator (message shown in red)
    - 20-25: System problems and fatal errors, and the client connection disconnected (message shown in red)


QATSQLPLUS

Working with Error Messages

- Errors stored in sys.messages
- Add new error messages can be added by using sp_add_message
- Replace placeholders in messages using FORMATMESSAGE()

Command outline:

```
EXEC sp_addmessage <error number> , <severity> , <message>

SELECT * FROM sys.messages

DECLARE @msg = FORMATMESSAGE(<error number>,<parameters...>)
```

Demonstration:

```
EXEC sp_add_message 50200,16,'Customer %d has been deleted by %s.'

SELECT * FROM sys.messages

DECLARE @msg = FORMATMESSAGE(50200, 2345, 'Fred Jones')
```

Working with Error Messages

- Errors stored
  - View all messages

    SELECT * FROM sys.messages
  - View custom error messages

    SELECT * FROM sys.messages WHERE message_id > 50000

- Add new error messages can be added by using sp_addmessage including placeholders
  - Placeholders:
    - %h : shortint
    - %l : longint
    - %d or %i : integer
    - %s : string

  - Adding new error messages

    EXEC sp_addmessage 50002, 6, 'The search returned no rows'

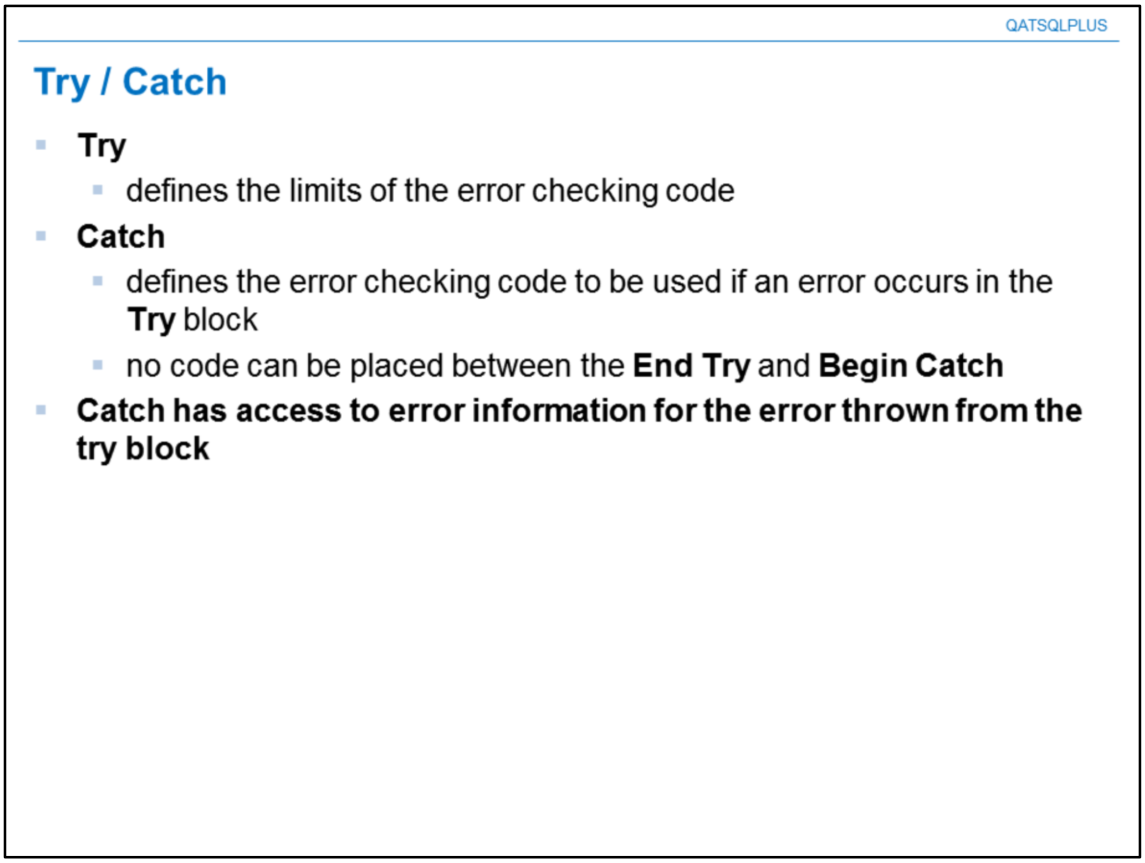
    EXEC sp_addmessage 50003,11, 'Execution of stored procedure %s cannot be completed as parameter %s was set to NULL'

- When placeholders can be replaced inline (when using RAISERROR) or using the FORMATMESSAGE() function before raising with RAISERROR or THROW (Raiserror and Throw are covered later in this module)
    - FORMATMESSAGE(messagenumber, replacement values in order) – where the error message has been added previously

```
DECLARE @ErrorMsg VARCHAR(30);
SET @ErrorMsg = FORMATMESSAGE(50003,'ProductID');
THROW 50003,@ErrorMsg,1
```

-
    - FORMATMESSAGE(messagetext, replacement values in order) – where the error message is created when required

```
DECLARE @ErrorMsg VARCHAR(30);
SET @ErrorMsg = FORMATMESSAGE('The table %s has been
changed since the view %s was
created','tbl_Source','tbl_ReadRows');
THROW 60000,@ErrorMsg,1
```


QATSQLPLUS

Try / Catch

- **Try**
  - defines the limits of the error checking code
- **Catch**
  - defines the error checking code to be used if an error occurs in the **Try** block
  - no code can be placed between the **End Try** and **Begin Catch**
- **Catch has access to error information for the error thrown from the try block**

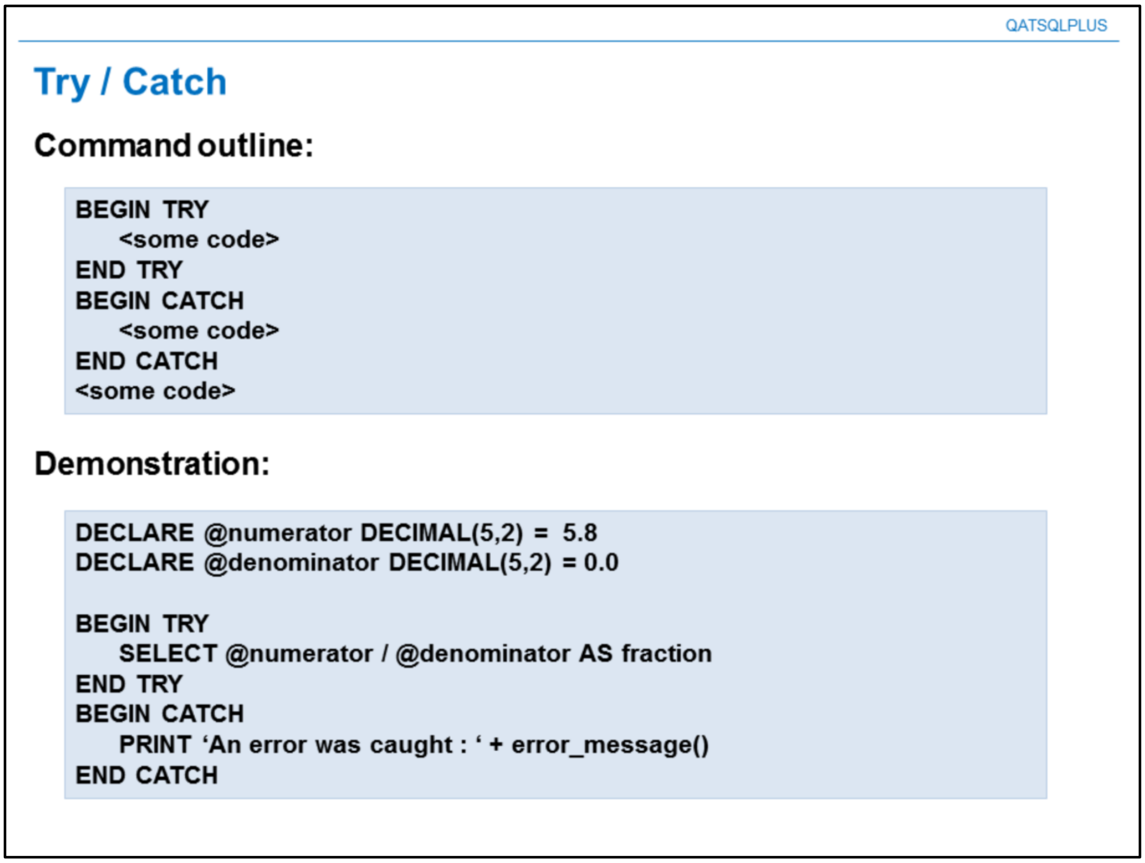
Try / Catch

- Try
  - defines the limits of the error checking code
  - BEGIN TRY defines the start
  - END TRY defines the end of the try block
  - Try block can be embedded

- Catch
  - Defines the error checking code to be used if an error occurs in the **Try** block
  - No code can be placed between the **End Try** and **Begin Catch**

```
BEGIN TRY
    …code
    BEGIN TRY
        ….code
    END TRY
    BEGIN CATCH
        ….code dealing with the inner TRY block
    END CATCH
END TRY
BEGIN CATCH
    ….code dealing with the outer TRY block
END CATCH
```

- Catch has access to error information for the error thrown from the try block:
  - Error_Message() – error message
  - Error_Number() – unique ID
  - Error_Severity() – severity of the error
  - Error_State() – whether the current transaction is recoverable
  - Error_Procedure() – the procedure that caused the error
  - Error_Line() – the line number within the procedure



QATSQLPLUS

Try / Catch

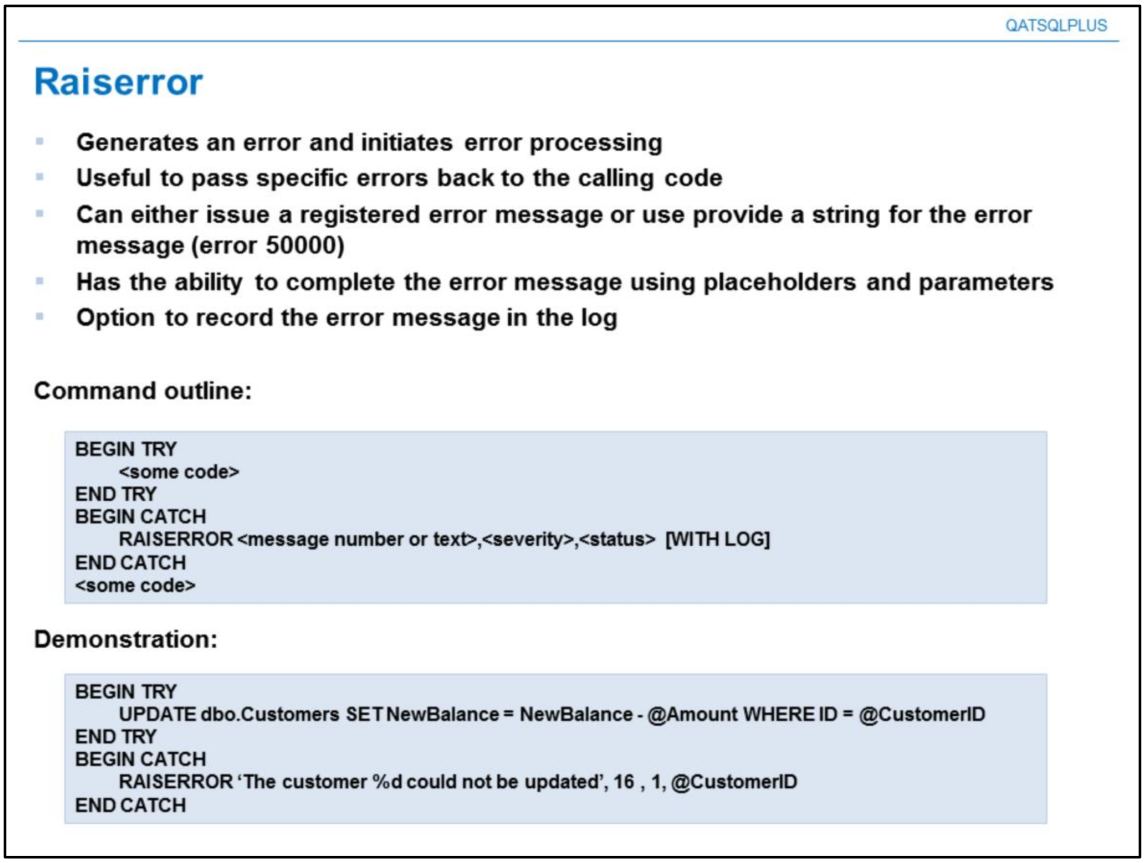
Command outline:

```
BEGIN TRY
    <some code>
END TRY
BEGIN CATCH
    <some code>
END CATCH
<some code>
```

Demonstration:

```
DECLARE @numerator DECIMAL(5,2) =  5.8
DECLARE @denominator DECIMAL(5,2) = 0.0

BEGIN TRY
    SELECT @numerator / @denominator AS fraction
END TRY
BEGIN CATCH
    PRINT ‘An error was caught : ‘ + error_message()
END CATCH
```


QATSQLPLUS

Raiserror

- Generates an error and initiates error processing
- Useful to pass specific errors back to the calling code
- Can either issue a registered error message or use provide a string for the error message (error 50000)
- Has the ability to complete the error message using placeholders and parameters
- Option to record the error message in the log

Command outline:

```
BEGIN TRY
      <some code>
END TRY
BEGIN CATCH
      RAISERROR <message number or text>,<severity>,<status>  [WITH LOG]
END CATCH
<some code>
```

Demonstration:

```
BEGIN TRY
      UPDATE dbo.Customers SET NewBalance = NewBalance - @Amount WHERE ID = @CustomerID
END TRY
BEGIN CATCH
      RAISERROR ‘The customer %d could not be updated’, 16 , 1, @CustomerID
END CATCH
```

Raiserror

- Generates an error and initiates error processing
    - If the error is raised with a try block then the catch block will catch the error and process it
    - If the error is raised outside a try block, then the calling application will be returned the error for processing
        - In the case of SSMS the error is displayed in the Messages tab
- Useful to pass specific errors back to the calling code
    - The calling application may be programmed to catch specific error numbers, so pre-defining these in the sys.messages will ensure they are available to the SQL Server
    - A message text could be used with error number 50000 instead

- Has the ability to complete the error message using placeholders and parameters
    - RAISERROR can either use placeholders in a added message or use the message text

Assuming error was created as:

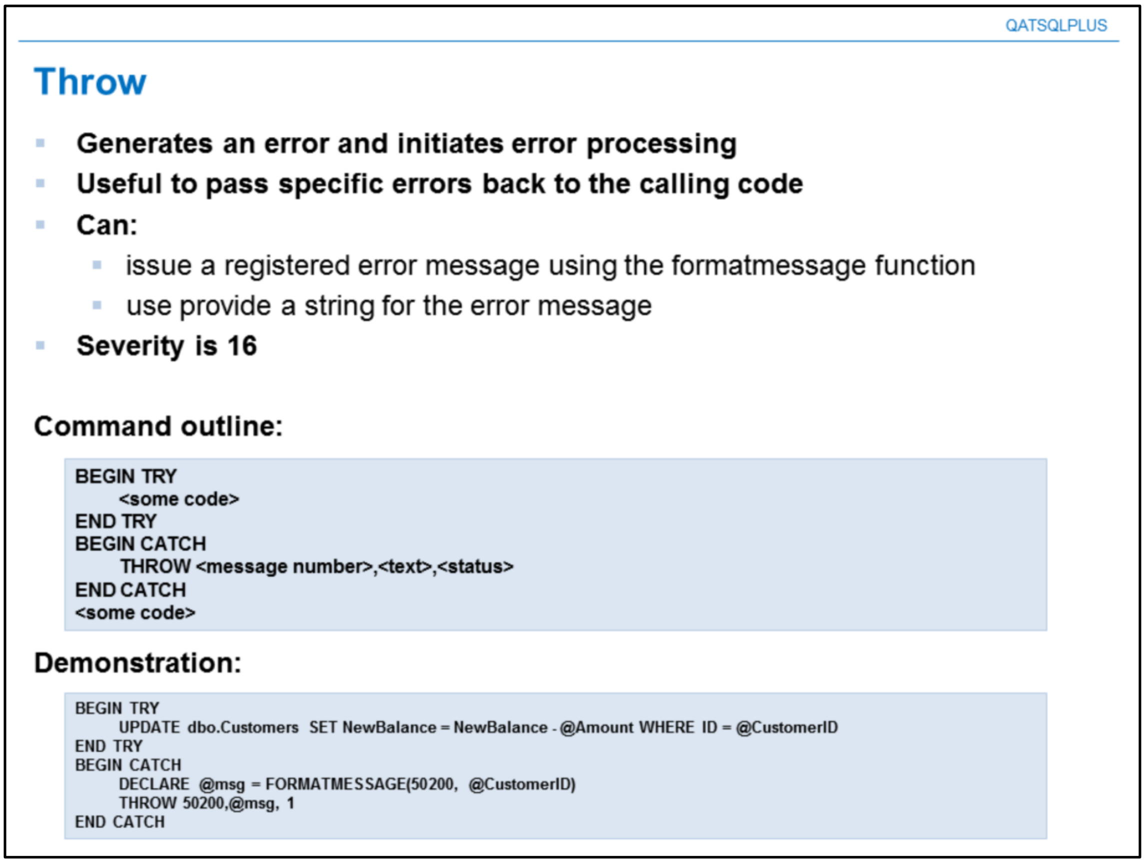
EXEC sp_addmessage 50090,14,’Error : not enough stock for product %s (only %d available and sale requires %d)’

RAISERROR using pre-created message ID:

RAISERROR (50090, 14, 1, ‘Apples’, 20, 100)

RAISERROR using message text
RAISERROR (’Error : not enough stock for product %s (only %d available and sale requires %d)’,14,1, ‘Apples’, 20, 100)

- Option to record the error message in the log by adding WITH LOG
- RAISERROR has a message in the MSDN help that THROW should be used for new application


QATSQLPLUS

Throw

- Generates an error and initiates error processing
- Useful to pass specific errors back to the calling code
- Can:
  - issue a registered error message using the formatmessage function
  - use provide a string for the error message
- Severity is 16

Command outline:

```
BEGIN TRY
      <some code>
END TRY
BEGIN CATCH
      THROW <message number>,<text>,<status>
END CATCH
<some code>
```

Demonstration:

```
BEGIN TRY
      UPDATE dbo.Customers SET NewBalance = NewBalance - @Amount WHERE ID = @CustomerID
END TRY
BEGIN CATCH
      DECLARE @msg = FORMATMESSAGE(50200, @CustomerID)
      THROW 50200,@msg, 1
END CATCH
```

Throw

- Similar in structure to RAISERROR but aligned to visual basic and c#, where errors are THROWN
- Throw always uses the error severity 16
- Generates an error and initiates error processing
  - If the error is raised with a try block then the catch lock will catch the error and process it
  - If the error is raised outside a try block, then the calling application will be returned the error for processing
    - In the case of SSMS the error is displayed in the Messages tab
- Useful to pass specific errors back to the calling code
  - The calling application may be programmed to catch specific error numbers, so pre-defining these in the sys.messages will ensure they are available to the SQL Server
  - A message text could be used with error number any number above 50000
- Throw cannot complete the error message using placeholders and parameters, but rather should use the FORMATMESSAGE function
  - Throw can either use placeholders in a added message or use the message text

Assuming error was created as:

EXEC sp_addmessage 50090,14,'Error : not enough stock for product %s (only %d available and sale requires %d)'

THROW using pre-created message ID:

DECLARE @ErrMsg VARCHAR(300)

SET @ErrMsg = FORMATMESSAGE(50090, 'Apples', 20, 100);

THROW 50090, @Errmsg, 1

THROW using message text

```
        DECLARE @ErrMsg VARCHAR(300)
        SET @ErrMsg = FORMATMESSAGE('Error : not enough stock for
product %s (only %d available and sale requires %d)','Apples', 20, 100);
        THROW 80000, @Errmsg, 1
```

QATSQLPLUS

Exercise